Enclosure A

SDK Overview

Welcome to the Flight Simulator 98 Software Development Kit. This reference is aimed at experienced developers who want to modify or extend Flight Simulator 98. Those attempting to use this SDK should be familiar with C++ programming conventions.

Because of the high demand for this information, this SDK is being distributed on the Microsoft Flight Simulator web site in sections, as quickly as possible. When the SDK is complete, it will contain the following SDK sections:

- Panels
- Aircraft Container
- Adventure Programming Language
- Scenery
- Multiplayer

The SDK is distributed as a Microsoft Windows Help file to facilitate updates, as well as its distribution and use. When the SDK is complete, an .rtf file version of the SDK will also be released.

Important: This SDK is not supported by Microsoft Product Support Services.

Related Topics

Panels SDK Overview

The Panels SDK includes everything you need to know to create, modify, and install fully functioning panels in Flight Simulator 98. You'll find the files that you need, complete with examples of existing files. You'll find source code that you can just cut and paste. You'll also find a programming reference that covers the structures and elements used in creating a panel.

If you want to create a gauge from scratch, take a look at the topics included in the Defining Panels and Defining Gauges sections. If you want to modify a gauge or understand the "ins and outs" of how a specific drawing element interacts with the Panel system, see the Drawing Elements section. For reference information, see the Programming Reference section, as well as the topics included in the rest of the SDK that focus on specific functions and how they interact with the Panel system.

Important: The information included in the Panels SDK is intended as a reference for programmers. It assumes familiarity with C programming language and game development. The information is not supported by Microsoft Product Support.

In Flight Simulator 98, all aircraft-specific files are located in the Aircraft folder, also called the container system. The aircraft folder includes a folder for each aircraft available in Flight Simulator 98. In turn, each individual aircraft folder includes the following folders and files:

- Model folder
- Panel folder
- Sound folder
- Texture folder
- Aircraft configuration file
- Checklists configuration file

The Panels SDK covers the Panel folder and the Aircraft configuration file, which are essential to creating a panel.

Related Topics

Aircraft Container SDK Overview

The intent of the Aircraft Container SDK is to give you a clear understanding of aircraft containers: their structure, the files they contain and where the files reside, and the features you can use to get the most out of the system; namely, aliasing and configuration sets. This SDK contains information on how to modify and share components among existing aircraft, it does not cover how to create new aircraft.

Important: The files described in the Aircraft Container SDK are simple text files (configuration, .cfg, files) and can be manipulated using any text editor, such as Microsoft Notepad. Please note that these files should only be modified with caution by experienced developers, as changes could render the aircraft inoperable.

The Aircraft Container system provides a structure of containment for all the files and attributes for aircraft in Flight Simulator 98. The on-disk structure is consistent and logical; all aircraft-related files are located close together. The key advantage of the Aircraft Container system is that it provides a framework that can represent several aircraft; by using the Aircraft Container system, you can avoid duplicating files. If your aircraft share common components, you only need to create those components once. Using aliasing and configuration sets, any aircraft can access any component in the Aircraft Container system. In short, the Aircraft Container system allows you to easily customize your aircraft and you can avoid duplicating files you've already created. The following folders contain aircraft-related files:

- Aircraft folder: Contains all the folders and files included in this list.
- Aircraft Container folders (for example, Bell206B): Contain folders and files that define aircraft such as visual models, textures, panels, and sounds, as well as the Aircraft.cfg file and the checklist .cfg files.
- Model folder: Contains the Model.cfg file and .mdl files.
- Panel folder: Contains the Panel.cfg file and .bmp files.
- Sound folder: Contains the Sound.cfg file and .wav files.
- Texture folder: Contains .r8 files.

Related Topics

APL SDK Overview

The Adventure Programming Language (APL) SDK is intended to give you an understanding of APL as a full-featured programming language that you can use to create and edit adventures for Flight Simulator. The APL SDK includes sections that cover:

- Language features specific to APL.
- The APL compiler (Aplc.exe).
- The adventure (.adv) file structure.
- The APL binary format.
- The interaction between APL and Flight Simulator.
- A complete APL language reference that documents command syntax, commands, expressions, functions, and Flight Simulator variables that are available to APL.
- Special subroutines and event-driven programming.

In short, the APL SDK gives you the information you need on the programming tools necessary to put it all together and create your own, custom adventures for Flight Simulator. The APL SDK .zip file, located on this web site, also includes a sample APL adventure that you can compile.

Important: The information included in the APL SDK is intended as a reference for programmers. It assumes familiarity with the C and BASIC programming languages. The information included in this SDK is not supported by Microsoft Product Support.

Related Topics

Scenery SDK Overview

The Scenery SDK includes explanations and examples of the concepts and elements used to design scenery for Flight Simulator. In particular, this SDK both explains and provides a programmer's reference for the BGL graphics language; it also explains (and provides examples of) the BGL files that contain scenery. The key to understanding scenery design in Flight Simulator lies in understanding the general concepts and conventions used throughout the graphics system, and then building on those basics. If you're not familiar with BGL, the following list suggests a route through the Scenery SDK that progresses from the general concepts used throughout BGL to more specific details.

- First, learn the basics about coordinate systems; you'll find the details in the topics included in the Coordinate Conventions section. These topics will familiarize you with the grid layout that the Flight Simulator 98 world and designs are based on.
- Look at the topics in the Seeds and Objects sections and learn the basics of seeds and objects. Seeds and objects make up the two categories of scenery design in BGL databases.
- Familiarize yourself with the BGL File Structure section. It's important to know the classes of entities that BGL uses in designs.
- Take a glance at the Exception Entities topic in the File Formats section. You don't need to know all the details to begin with, but it's important to know that these features exist and to understand the key concepts behind them.
- To get an idea of what the actual scenery files are and where they should go on the hard disk, refer to the Graphics Databases and Scenery File Location and Data Flow sections.
- Go through the topics in the Basic Design Techniques, Examples. This section is where it all comes together, and going through the examples in these topics will give you the practical working knowledge you need to design scenery using BGL. All of the concepts outlined in the rest of the SDK are put into play here, and actual databases are created. You can use these examples to set up your own design files and duplicate the results in your own development environment.

The other sections in the SDK are important, too. You'll find sections that include more advanced topics dealing with the specifics and subtleties of scenery design, such as the section Hazing, Shadows, and Special effects. You'll also find the BGL Language Reference; a great resource that you can refer to on an as-needed basis.

Important: The information included in the Scenery SDK is intended as a reference for programmers. It assumes familiarity with C programming language, Macro Assembler (MASM), and game development. The information is not supported by Microsoft Product Support.

The following sections describe the language, syntax, and types of math used throughout this SDK.

Use of Assembly Language and C Syntax

Simple assembly language syntax is used to define the arrangement of bytes and words of data within databases. Any MASM reference manual describes the meaning of this syntax.

C syntax is also used occasionally. The use of both C and assembly language syntax doesn't require any proficient knowledge of x86, Pentium assembly language, C, or C++. These syntaxes are used to show how data structures are laid out to create graphics databases.

Hexadecimal notation is used extensively. Values that are in "hex" are indicated in assembly language format: 1234h and 0b7f2h. If a hex value begins with a letter value, a zero always precedes it.

Use of Integer and Fractional Math and its Notation

To make the best use of computer hardware, Flight Simulator represents values in integer format to use the processor's integer math units. Although fast floating point math units make mathematical operations fast regardless or integer or floating point types, moving 1, 2, and 4-byte integer pieces around in memory and registers is still faster. Integers are used to represent both whole numbers and fractions.

Whole Number Integer Math

The smallest integer unit that is dealt with is the byte. In many instances, an integer byte is represented with the letter "i". This value has a range of -128–127 if it is signed and 0–255 if it is unsigned.

An integer word consists of 2-bytes and is often referred to as an "ii" data type. This data type ranges from 0–65535 if it is unsigned and -32768–+32767 if it is signed.

Fractional Integer Math

Fractional values of less than one are dealt with by defining the values as fractional, using the integer processing units in the microprocessors.

Although a whole number integer word ranging from 0–65535 may seem like just "whole numbers," there is actually a binary point to the right of the least significant bit in the integer. The weights of the bits to the left of the binary point are 1, 2, 4, 8 and so on all the way up to bit 15, which is 32768.

Fractional numbers are created by putting bits to the right of the binary point. The first bit to the right has the weight of 1/2, the next 1/4, then 1/8, 1/16, 1/32, and so on. A fractional byte is represented by the letter "f". This byte has a range of .00h–.ffh or from 0–255/256 in increments of 1/256ths. A fractional word consists of 2 fractional bytes and is represented by "ff".

Mixed Whole Number and Fractional Math

Integers are often made to represent both whole number and fractional parts. For example, a 32-bit integer can represent 16 bits whole number, and 16 bits fractional. The binary point is in the middle of the 32 bits. The notation for this type of number is, "ii.ff".

Related Topics

Multiplayer SDK Overview

Important: The following is a discussion of the Flight Simulator Multiplayer system. An understanding of Microsoft's Direct Play technology (from the Direct-X 5 SDK) is recommended before reading this document. The DirectX 5 SDK is available on Microsoft's web site at http://www.microsoft.com/directx.

Earlier versions of Flight Simulator (prior to Flight Simulator, version 6.0) contained a multiplayer system that allowed users to connect two machines together so they could fly in a shared airspace. When Flight Simulator, version 6.0 was ported to the Windows platform, the multiplayer functionality was lost. This was fixed when a multiplayer system was added to Flight Simulator, version 6.1. The goal of the Flight Simulator, version 6.1 multiplayer system was to leverage existing Microsoft technology (DirectPlay) to produce a multiuser environment where multiple users could share the same airspace over a modem. The result was the multiplayer system found in Flight Simulator, version 6.1. The system will support multiple people via multiple protocols in a peer-to-peer gaming environment. An interface to the Internet Gaming Zone was provided so people can readily find other pilots from around the world.

The multiplayer system found in Flight Simulator, version 6.1 is built on Microsoft's DirectPlay technology , and doesn't provide external hooks for access to data relating to the remote players in a session. It is possible to write an application that will hook into the message stream of an active multiplayer session to gain access to the remote player data.

At the core of the multiplayer system we have the players. Presently, players fall into three categories: unknown, players and observers. The unknown player designation is a temporary designation used when a player first joins a session. A player will be classified as unknown until they receive a response from the session host to see if they can join as a player or observer. The player classification is used to designate a player that has a visual presence on remote machines, and therefore can be collided with. The observer classification is used to designate a player that watches other players, but has no visual presence and cannot collide with other players.

To make the system function, information packets are sent between the players in a session. Each of these packets has a unique name and ID associated with it. Some packets also contain extra information and will have a data structure associated with them. To reduce the verbosity of the documentation, the packet ID and structure names for the packets will be referred to by the endings of their names. For packet ids, the **MULTIPLAYER_PACKET_ID_** prefix will be omitted, and for packet structures, the **MULTIPLAYER_PACKET_** prefix will be omitted. Complete lists of the packet IDs and packet structures used by the multiplayer system can be found in the topics Packet IDs and Packet Structures, respectively.

The multiplayer system provides a foundation for getting small groups of users in a single airspace. There is enough flexibility to allow the users to create their own experience.

Related Topics